

# Kafka Módulo I



**Igor Maldonado Floôr**

**Líder de desenvolvimento @Monitora**

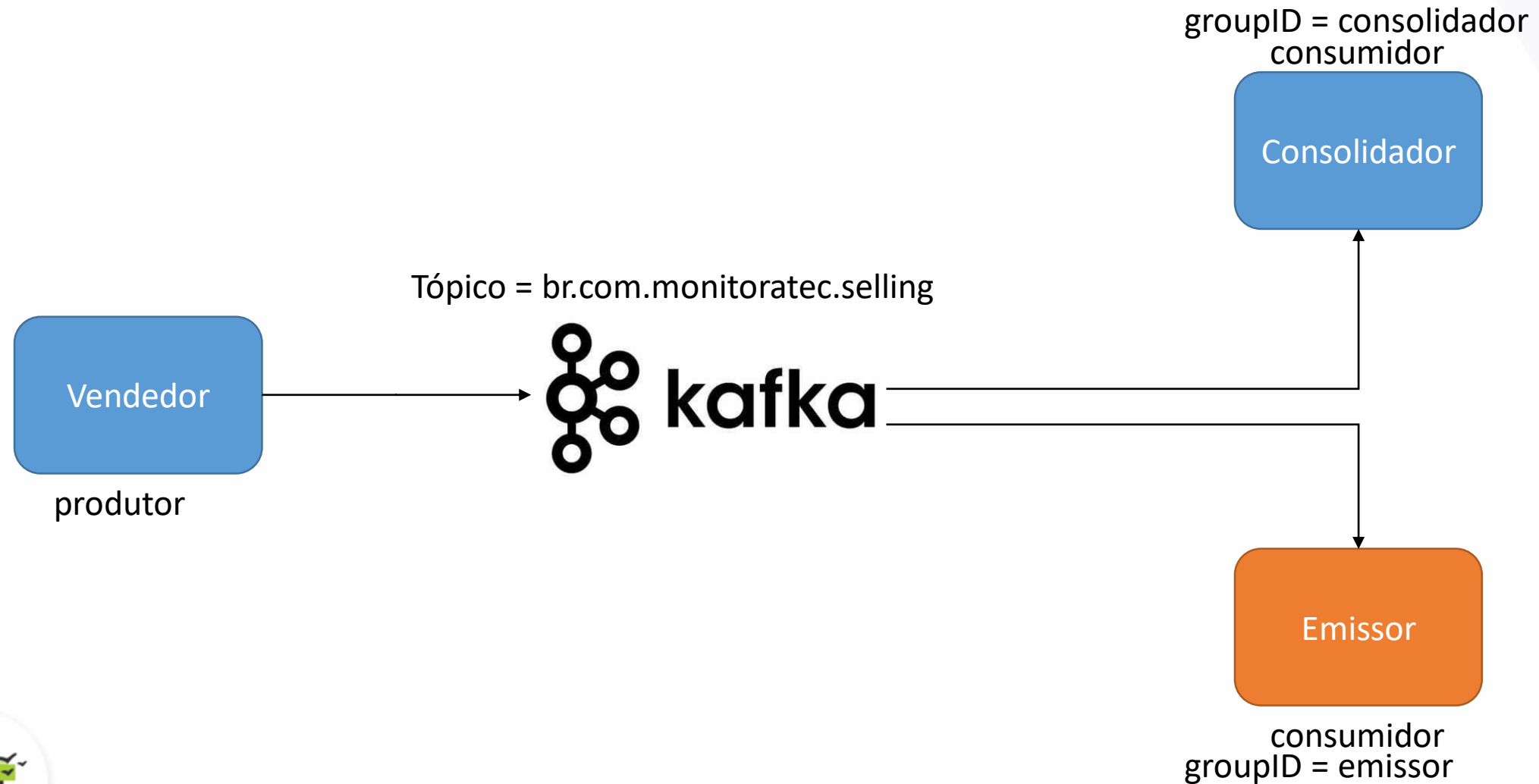
# Cronograma

---

Dia	Objetivo
1 – 27/07	Conceitos iniciais de Kafka
2 – 29/07	Conceitos de funcionamento do Kafka
3 – 03/08	Hands on: setup inicial + produtores em java
4 – 05/08	Hands on: Consumidores em java

# Hands on 2

# Cenário live



# Desenvolvimento consumidor

# Passos

---

1. Projeto inicial
2. Instalar dependências (maven)
3. Criação do model Selling (avro)
4. Configurações de propriedades (properties.yml)
5. Implementação Kafka consumer + configs
6. Desenvolvimento do consumidor



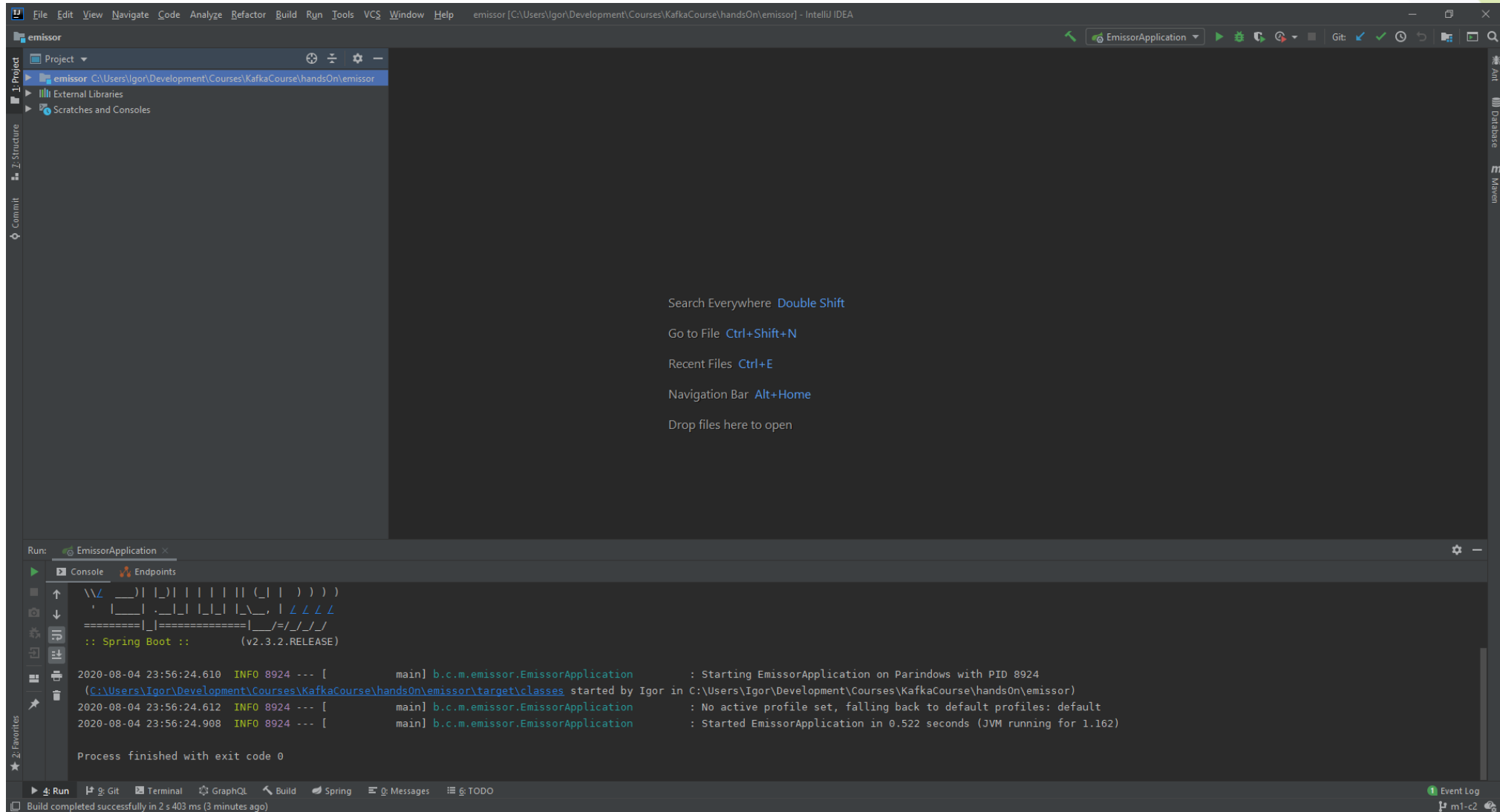
## Step 1 - Projeto inicial

---

- Branch: m1-c1
- \$GIT/handsOn/emissor
- Executar projeto vazio



# Step 1 - Projeto inicial





## Step 2 - Instalar dependências

---

- Trabalhar no arquivo pom.xml
- Ele está na raiz do projeto



## Step 2 - Instalar dependências

---

```
<properties>
  <java.version>12</java.version>
</properties>

<repositories>
  <repository>
    <id>confluent</id>
    <url>http://packages.confluent.io/maven/</url>
    <releases>
      <enabled>true</enabled>
    </releases>
  </repository>
</repositories>

<dependencies>
```



## Step 2 - Instalar dependências

---

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.kafka</groupId>
    <artifactId>spring-kafka</artifactId>
  </dependency>
  <dependency>
    <groupId>io.confluent</groupId>
    <artifactId>kafka-avro-serializer</artifactId>
    <version>5.5.0</version>
  </dependency>

  <dependency>
    <groupId>org.apache.avro</groupId>
    <artifactId>avro</artifactId>
    <version>1.9.2</version>
  </dependency>
  <dependency>
```



## Step 2 - Instalar dependências

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
    <plugin>
      <groupId>org.apache.avro</groupId>
      <artifactId>avro-maven-plugin</artifactId>
      <version>1.9.2</version>
      <executions>
        <execution>
          <phase>generate-sources</phase>
          <goals>
            <goal>schema</goal>
          </goals>
          <configuration>
            <sourceDirectory>${project.basedir}/src/main/resources/avro-sources/</sourceDirectory>
            <outputDirectory>${project.basedir}/src/main/java/br/com/monitoratec/vendedor/</outputDirectory>
            <stringType>String</stringType>
          </configuration>
        </execution>
      </executions>
      <configuration>
        <imports>
          <!-- Common -->
          <!-- seDTO.avsc</import>-->
          <import>${project.basedir}/src/main/resources/avro-sources/common/commonResponseDTO.avsc</import>-->
        </imports>
      </configuration>
    </plugin>
  </plugins>
</build>
```



## Step 2 - Instalar dependências

---

- Instalar dependências pelo maven
- Branch m1-c2



## Step 3 - Criação do model Selling (avro)

---

- Pasta de resources: avro-sources
- Arquivo: selling.avsc



## Step 3 - Criação do model Selling (avro)

```
selling.avsc x
1  {
2    "type": "record",
3    "name": "Selling",
4    "namespace": "kafka.avro.generated",
5    "fields": [
6      {
7        "name": "amount",
8        "type": "double"
9      },
10     {
11       "name": "buyer",
12       "type": "string"
13     }
14   ]
15 }
```

## Step 3 - Criação do model Selling (avro)

---

- Executar: mvn package (pode ser pela UI)
- Verificar pasta: sources: kafka/avro/generated
- Branch m1-c3

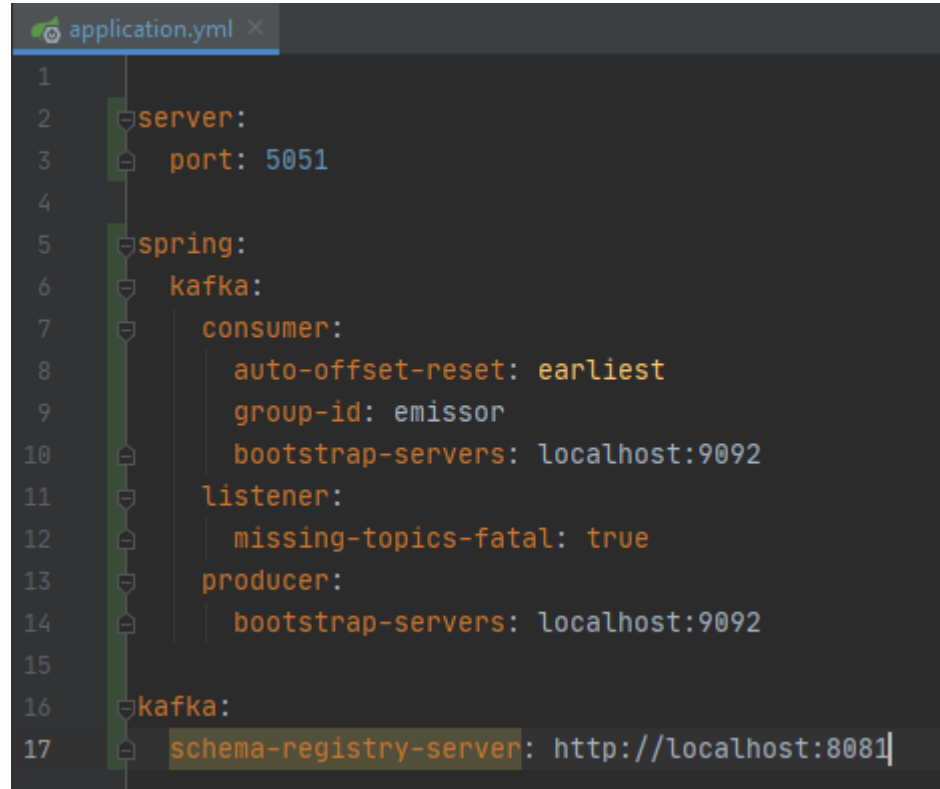




## Step 4 - Configurações de propriedades

---

- Configuração de propriedades da aplicação (Spring app)



```
1
2  server:
3    port: 5051
4
5  spring:
6    kafka:
7      consumer:
8        auto-offset-reset: earliest
9        group-id: emissor
10       bootstrap-servers: localhost:9092
11      listener:
12        missing-topics-fatal: true
13      producer:
14        bootstrap-servers: localhost:9092
15
16     kafka:
17       schema-registry-server: http://localhost:8081
```

## Step 4 - Configurações de propriedades

---

- Arquivo está localizado em: resources: application.yml
- Branch m1-c4



## Step 5 - Implementação Kafka consumer + configs

---

- Implementação da configuração de consumidores do projeto
- Pasta: Kafka/config



## Step 5 - Implementação Kafka consumer + configs

```
KafkaAvroDeserializerErrorHandled.java
1  package br.com.monitoretec.emissor.kafka.config;
2
3  import io.confluent.kafka.schemaregistry.client.SchemaRegistryClient;
4  import io.confluent.kafka.serializers.KafkaAvroDeserializer;
5  import org.apache.avro.Schema;
6  import org.slf4j.Logger;
7  import org.slf4j.LoggerFactory;
8
9  import java.util.Map;
10
11  public class KafkaAvroDeserializerErrorHandled extends KafkaAvroDeserializer {
12      private static final Logger log = LoggerFactory.getLogger(KafkaAvroDeserializerErrorHandled.class);
13
14      public KafkaAvroDeserializerErrorHandled(SchemaRegistryClient client) { super(client); }
15
16      public KafkaAvroDeserializerErrorHandled(SchemaRegistryClient client, Map<String, ?> props) {
17          super(client, props);
18      }
19
20      @Override
21      public Object deserialize(String s, byte[] bytes) {
22          try {
23              return super.deserialize(s, bytes);
24          } catch (Exception e) {
25              log.error("Error on avro deserialization", e);
26              return null;
27          }
28      }
29
30      @Override
31      public Object deserialize(String s, byte[] bytes, Schema readerSchema) {
32          try {
33              return super.deserialize(s, bytes, readerSchema);
34          } catch (Exception e) {
35              log.error("Error on avro deserialization", e);
36              return null;
37          }
38      }
39  }
```



## Step 5 - Implementação Kafka consumer + configs

```
KafkaConsumer.java
17 import java.util.Map;
18
19 import static io.confluent.kafka.serializers.AbstractKafkaSchemaSerDeConfig.SCHEMA_REGISTRY_URL_CONFIG;
20
21 @EnableKafka
22 @Configuration
23 public class KafkaConsumer {
24
25     @Value("${spring.kafka.consumer.bootstrap-servers}")
26     private String bootstrapServers;
27
28     @Value("${kafka.schema-registry-server}")
29     private String schemaRegistryServer;
30
31     @Value("${spring.kafka.consumer.group-id}")
32     private String groupId;
33
34     public KafkaConsumer() {}
35
36     public Map<String, Object> consumerConfigs() {
37         Map<String, Object> configProps = new HashMap<>();
38         configProps.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapServers);
39         configProps.put(ConsumerConfig.GROUP_ID_CONFIG, groupId);
40         configProps.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class);
41         configProps.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, KafkaAvroDeserializer.class);
42         configProps.put(KafkaAvroDeserializerConfig.SPECIFIC_AVRO_READER_CONFIG, "true");
43         configProps.put(SCHEMA_REGISTRY_URL_CONFIG, schemaRegistryServer);
44         return configProps;
45     }
46
47     public ConsumerFactory<String, Object> consumerFactory() {
48         try {
49             (KafkaAvroDeserializerErrorHandled kafkaAvroDeserializer = new KafkaAvroDeserializerErrorHandled(new CachedSchemaRegistryClient(schemaRegistryServer, Integer.MAX_VALUE), consumerConfigs())) {
49                 return new DefaultKafkaConsumerFactory<>(consumerConfigs(), new StringDeserializer(), kafkaAvroDeserializer);
50             }
51         }
52     }
53
54     @Bean
55     public ConcurrentKafkaListenerContainerFactory<String, Object> kafkaListenerContainerFactory() {
56         ConcurrentKafkaListenerContainerFactory<String, Object> factory = new ConcurrentKafkaListenerContainerFactory<>();
57         factory.setConsumerFactory(consumerFactory());
58         factory.setMissingTopicsFatal(true);
59         return factory;
60     }
61 }
```



## Step 5 - Implementação Kafka consumer + configs

---

- Branch m1-c5



## Step 6 - Desenvolvimento do consumidor

---

- Criaremos a configuração inicial do tópico que iremos ouvir
- Adição do EmitterListener : sources: listeners/  
EmitterListener.java



## Step 6 - Desenvolvimento do consumidor

---

```
EmissorListener.java x
1  package br.com.monitoratec.emissor.kafka.listeners;
2
3  import ...
4
5
6
7  @Component
8  public class EmissorListener {
9
10     @Bean
11     public NewTopic adviceTopic() {
12         return new NewTopic( name: "br.com.monitoratec.selling", numPartitions: 3, (short) 1);
13     }
14 }
15 |
```



## Step 6 - Desenvolvimento do consumidor

```
EmissorListener.java x
1 package br.com.monitoratec.emissor.kafka.listeners;
2
3 import kafka.avro.generated.Selling;
4 import org.apache.kafka.clients.admin.NewTopic;
5 import org.springframework.context.annotation.Bean;
6 import org.springframework.kafka.annotation.KafkaListener;
7 import org.springframework.stereotype.Component;
8
9 @Component
10 public class EmissorListener {
11
12     @KafkaListener(topics = "br.com.monitoratec.selling")
13     public void listen(Selling selling) {
14         System.out.println("Received selling, buyer: [" + selling.getBuyer() + "], amount: [US$ " + selling.getAmount() + "]);
15         System.out.println("Emitting coupon...\n");
16     }
17
18     @Bean
19     public NewTopic adviceTopic() { return new NewTopic( name: "br.com.monitoratec.selling", numPartitions: 3, (short) 1); }
20
21 }
22
23
```



## Step 6 - Desenvolvimento do produtor

---

- Branch m1-c6





# Post hands on

## Post hands on

---

- Verificar mensagens no Broker (via Kafka ui)
- localhost:3030
- Rodar 2 emissores





# Dúvidas?



Passeio São Carlos - Av. Dr. Francisco Pereira Lopes, 1701, Lojas 17 e 18  
Parque Santa Mônica, São Carlos/SP - CEP: 13.564-002

(16) 3419-7100 / (16) 3419-7200  
[www.monitoretec.com.br](http://www.monitoretec.com.br)